# A scalable 3D HOG model for fast object detection and viewpoint estimation

Marco Pedersoli Tinne Tuytelaars

KU Leuven, ESAT/PSI - iMinds
Kasteelpark Arenberg 10 B-3001 Leuven, Belgium
firstname.lastname@esat.kuleuven.be

## Abstract

*In this paper we present a scalable way to learn and detect objects using a 3D representation based on HOG patches placed on a 3D cuboid. The model consists of a single 3D representation that is shared among views. Similarly to the work of Fidler* et al. *[5], at detection time this representation is projected on the image plane over the desired viewpoints. However, whereas in [5] the projection is done at image-level and therefore the computational cost is linear in the number of views, in our model every view is approximated at feature level as a linear combination of the pre-computed fronto-parallel views. As a result, once the fronto-parallel views have been computed, the cost of computing new views is almost negligible. This allows the model to be evaluated on many more viewpoints.*

*In the experimental results we show that the proposed model has a comparable detection and pose estimation performance to standard multiview HOG detectors, but it is faster, it scales very well with the number of views and can better generalize to unseen views. Finally, we also show that with a procedure similar to label propagation it is possible to train the model even without using pose annotations at training time.*

## 1. Introduction

Object detection is now a mature field [3]. Generally, objects observed from a *specific point of view* have a similar appearance and modern detectors based on a 2D representation of the object like deformable part models (DPM) [4] can detect most of the object instances with high recall and few false positives. However, objects in the real world are three dimensional, therefore their appearance can drastically change depending on the camera viewpoint. Consequently a single 2D model is not enough and leads to poor performance. To address this problem the most common solution is to sample the most important views of the model and then use a 2D model for each view.

In this paper we argue that such approach based on sampled 2D views is sub-optimal and it is better to use a 3D model of the appearance of the class. In more detail, a model based on sampled 2D views is sub-optimal because: (i) views of the object that are far from the modeled views are difficult to detect, (ii) increasing the number of modeled views reduces the number of training samples for each view, and (iii) the computational cost of the model at test time grows linearly in the number of views and can easily become untractable, especially if we go beyond a 1D viewing circle and model the full viewing sphere.

With our method we build a 3D representation of the object class and therefore we solve or alleviate at once all the previously mentioned problems. As shown in Fig. 1 for the car class, we propose to approximate the 3D appearance of an object with a cuboid representation, where each face of the cuboid is composed of a set of HOG patches whose appearance is learned during training. The detection for a given object viewpoint is then effectuated as an orthogonal projection of the 3D model on the 2D image. With this model then: (i) we can detect any possible viewpoint of the object because in contrast to [10, 12] the model is not a composition of 2D views but a real 3D model (sect. 3) (ii) all samples are represented by a single 3D appearance model, so the entire training data is shared among the different faces of the 3D model e.g. when during training a car is seen at 45 degrees orientation, this view is used to update both the lateral and the frontal faces of the model, (iii) in contrast to [5], we do not evaluate the model patches for every possible view, instead we evaluate them only for the views facing the camera (fronto-parallel) because the other views can be easily and quickly approximated as linear combinations of that view (sect. 4). In sect. 5 we explain how to learn the model in a fully supervised and a weakly supervised setting. An experimental evaluation of the different characteristics of our model and a comparison with previous approaches is presented in sect. 6, while conclusions are drawn in sect. 7.

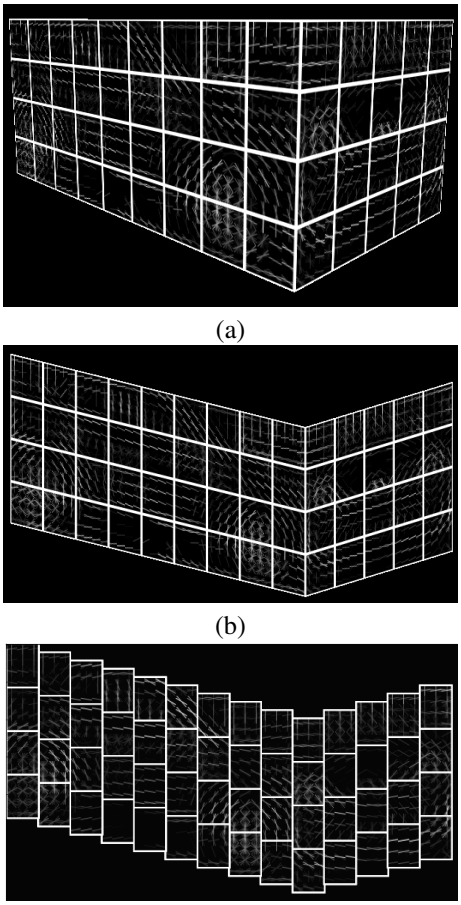


Figure 1. 3D object representation learned on epfl car dataset. (a) 3D representation at 45 degrees viewpoint, (b) 3D representation with orthogonal projection, (c) 2D approximation as a collection of 2D HOG filters.

## 2. Related work

During the last few years the need for a 3D representation for recognition has become clear and several ideas have been proposed. A seminal work on 3D object modeling is the work of Kushal *et al*. [9], where the 3D appearance of an object instance is reconstructed as a set of 3D textured patches and used to improve recognition. We do something similar, but not for a specific instance of an object, but for an object class which makes the problem more challenging.

Thomas *et al*. [14] use an implicit shape model scheme together with multi-view activation links that share features over different view-models for object class recognition. A similar approach based on implicit 3D shape models is presented in [1]. Hoiem *et al*. [8] learn a rough 3D object model based on a CRF part based segmentation that shares appearance among close views. An explicit modeling of multiple views (as well as affine distortions) is presented by Vedaldi and Zisserman [15]. In this case the model is based on HOG features and the views are independent but learned as latent variables of the same model. These methods show some improvements in terms of detection but do not evaluate their performance for pose estimation. The first thorough evaluation for pose estimation is presented in [10], where the authors show that learning 1-vs-all view specific deformable part models (DPM) [4] is a simple yet well performing way to estimate the object viewpoint. A more principled solution based on an extension of DPM to 3D models is presented in [12]. The main difference with standard DPM models is the supervision at part locations that become 3D landmarks. This can provide a better viewpoint estimation but at the cost of much more supervision. Similar in spirit is also the work of Hejrati *et al*. [7] where they detect and localize 3D points of cars with a mixture of trees of parts.

A different approach is proposed by Su et al. [13], where a part-based probabilistic representation based on local features is used to detect the object and estimate the 3D viewpoint. Even though the proposed representation is quite general and well defined, the performance of the method is actually quite low.

Although the main basic block of our model are HOG features learned based on a latent SVM framework, the way of modeling an object is very different from DPM. In DPM each object is decomposed into 2D independent views, and for each view parts are allowed to move with respect to their anchor point to approximate local changes in the structure of the object. We argue that most of the local displacements of the parts are due to viewpoint variations (that is why DPM generally works much better than a rigid HOG representation on quite rigid objects like bicycles, cars, etc..). Therefore, in our model, instead of trying to consider part displacements as noise, and then model it as independent gaussian distributions (given the object center) as in DPM, we model the part location and appearance as a 2D projection of the underlying 3D model. We can consider DPM and our model as the extreme cases of a more general representation. DPM assumes a very loose deformation model, where each part displacement is modeled independently and relies on a good part appearance model to avoid false positive detections. In contrast our model is very constrained because it assumes that part movement is due only to the 3D viewpoint. We do not model other possible sources of part displacements, like a non-rigid object deformation or different location of object parts on different object instances. We leave as future work further investigation on possible combinations of the two representations.

The 3D object detector proposed by Fidler *et al*. [5] is probably the most similar to ours. In their work the object model is represented as a cuboid and at detection time, for each face of the cuboid the image is distorted based on the sought viewpoint. The main disadvantage of the method is its high computational cost: for each view, the entire

pipeline of distorting the image, computing features and scanning the object faces is repeated so that the total cost of detection is linear in the number of viewpoints. In practice, scanning the image with all possible viewpoints becomes too slow. In the paper some a priori knowledge about the orientation of the object is used to avoid the evaluation of all viewpoints, but this limits the general applicability of the method to very specific conditions. In contrast, in our approach, instead of distorting the image to model the 3D viewpoint, we distort the object model, which makes inference much faster (see sect. 4) and allows the detector to search over all possible viewpoints.

Finally, another interesting work is proposed by Yu and Savarese [16], where they estimate the main 3D planes of an object for improving detection and pose estimation. Also in this case the 3D distortion of the object parts due to perspective deformation is done at image level which makes the method computationally expensive.

## 3. From 2D to 3D models

We represent the object model in 3 dimensions as a set of object patches, where each patch $i$ is localized by a 3D location $l_i = (l_x, l_y, l_z)$ and orientation $n_i = (n_x, n_y, n_z)$. The orientation is defined as the normal vector of the patch. The appearance of the patch is based on HOG features and is learned during training.

For simplicity we represent our object as a predefined cuboid whose dimensions are estimated from the aspect ratio of the bounding boxes in the training data (however, the representation is general enough to represent more complex shapes where each patch can have a different orientation like spheres or cones). That is, we split the training data based on the bounding box aspect ratio into 3 clusters and select the cluster with minimum aspect ratio as frontal view and the one with maximum one as lateral view. We could also use a more class specific representation of our 3D model, using for instance 3D CAD models as in [12]. However, this will make the model tuned for a specific class, whereas we want to be able to apply the model to any general object class without the need of an accurate 3D model.

During inference, given a certain object viewpoint defined as three angles $p = (\theta_x, \theta_y, \theta_z)$, each 3D part is converted to a 2D projection as shown in Fig. 1 (c). This consists of two steps: estimating for each part its location on the 2D plane and its 2D appearance. The location of the part on the 2D plane $(x^p, y^p)$ is computed as:

$$\begin{bmatrix} x^p \\ y^p \end{bmatrix} = P^L \left( R_p \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right), \tag{1}$$

where, $R_p$ is the rotation matrix composed of the sequential rotation over $\theta_x, \theta_y$ and $\theta_z$. For the sake of simplicity we assume an orthographic projection ($P^L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$). For objects far enough from the camera this representation is a good enough approximation of the object viewpoint. Moreover, this avoids the need to estimate any camera parameter. In this way we can work with a dataset of images coming from different unknown cameras, as it is generally the case in object detection. Considering that also the 3D structure of the model is still a coarse approximation of the real 3D shape, using a better projection model would not really help: small distortions in the object representation are absorbed by the HOG representation that is robust to small translations.

Thus, the global score of a certain 3D location $d = (d_x, d_y, d_z)$ and viewpoint $p$ is:

$$G(d, p) = \sum_{i \in \mathcal{P}} S(P^L(R_p(d + l_i))), \tag{2}$$

where $\mathcal{P}$ is the set of the parts of the object model. The obtained location $(x^p, y^p)$ of the object part for a certain viewpoint $p$ is in general non-integral. Instead of just selecting the closest integer location as in [2], we compute the score as a bilinear interpolation of the 4 closest locations $\mathcal{C}$:

$$S(x^p, y^p) = \sum_{c \in \mathcal{C}} S^I(x^c, y^c) \sqrt{(|x^p - x^c| - 1)^2 + (|y^p - y^c| - 1)^2}, \tag{3}$$

where $S^I(x^c, y^c)$ is the score of the $i^{th}$ part:

$$S^I(x^c, y^c) = P^A(M^i, p, n_i, I) \tag{4}$$

where $M^i$ is a vector of weights associated to the model, $I$ is the current image and $P^A$ is a transformation that takes into account the appearance distortion due to a non-frontal view of a part. In [5] the distortion on appearance $P^I$ is applied to the image, such that:

$$S^I(x^c, y^c) =< M_i, H(P^I(p, n_i, I) >, \tag{5}$$

with $H(I)$ the HOG features computed on the image $I$. This formulation gives satisfactory results, but it is quite slow because for each pose new HOG features $H$ have to be computed and evaluated. In the next section we propose a much faster solution.

## 4. Fast filters approximation

Instead of applying the appearance distortion at image level as in Eq.(5), we apply it at model level. That is, instead of applying a distortion $P^I$ to the image, we apply the inverse distortion $P^M$ to the model:

$$S^I(x^c, y^c) =< P^M(M_i, p, n_i), H(I) > . \tag{6}$$

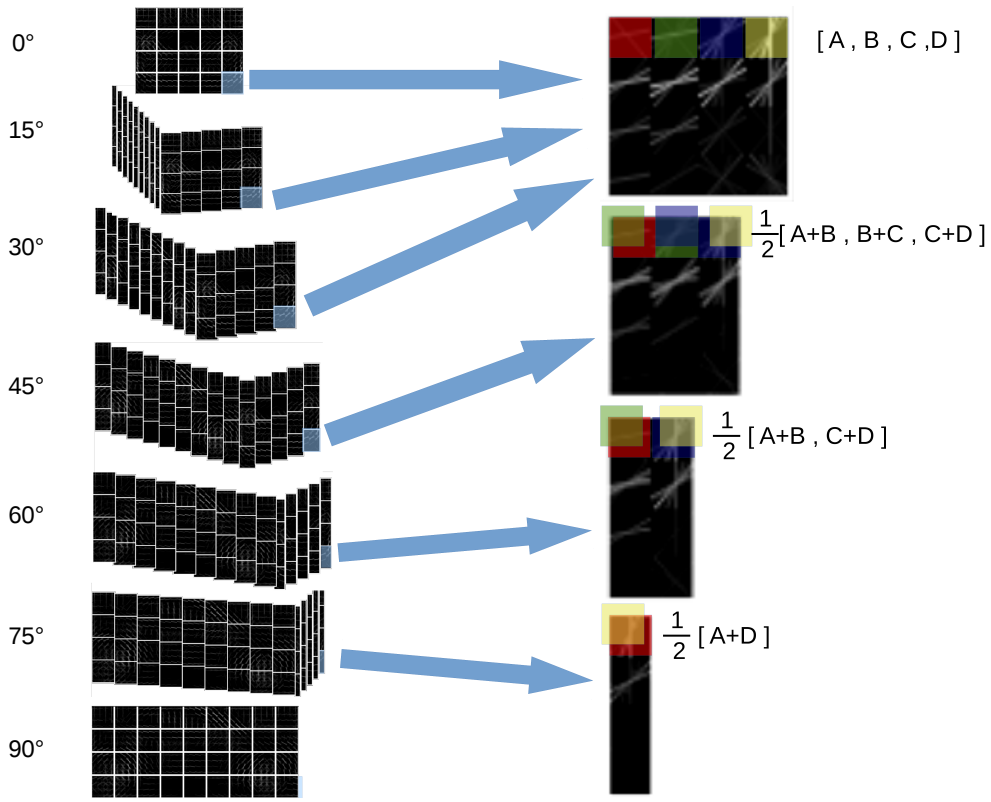



Figure 2. Filters approximation for different angles. Depending on the angle under which a part is visualized we can use the frontal filter (up to 30 degrees), or filters that are the linear combination of the HOG cells composing the frontal filter (in figure A,B,C,D). In this way it is not necessary to recompute the filters from scratch for different angles. For angles in between the shown values the closest approximation is chosen. Notice that the filter approximation is hand-crafted to reduce as much as possible the quantization effect.

In practice, as we deal with orthogonal projections, the distortion to apply to the model is an affine transformation. Assuming that we do not consider in plane rotations and ignoring the skew on a local patch scale, the distortion is a horizontal or vertical shrinking of each model patch. This can be approximated as a linear combination of the score generated by each HOG cell model. In Fig. 2 we show the filter approximations that have been used for each patch angle, which is computed based on the object viewpoint $p$ and the patch orientation $n_i$ (in the figure set to $(0, 0, 1)$).

The score of a part $i$ for a certain patch orientation $n_i$ and a given object viewpoint $p$ is computed as:

$$< P^M(M_i, p, n_i), H(I) >= \sum_{x=1,F_x} \sum_{y=1,F_y} D(M_i, p, n_i)(x, y)H(x, y), \quad (7)$$

where $F_x, F_y$ are the number of HOG cells of the patch and $D$ is a hand-crafted linear combination of the HOG cells as shown in Fig. 2. As the score is computed as a linear combination of different HOG cell scores, we can cache the score of each model cell $M_i(x, y)$ for each HOG cell location. As the computation of the score of a HOG cell involves 31 basic operations, it is much more expensive than summing the pre-computed scores. In practice, while computing a HOG cell score has a cost of 31, the subsequent linear combination has a cost of only 2 operations. Thus, after the initial caching, the cost of computing any possible orientation filter is around 16 times smaller and no further features or filtering are needed, producing an impressive computational saving.

## 5. Learning

For learning our 3D model we use latent SVM [4], where the viewpoint of the object is given. Still, we let the algorithm find the best location and scale in the image where to localize each training sample. Thus scale and location of the object are latent variables. To avoid to select locations in the background we enforce the latent assignment to have a minimum overlap (in the sense of intersection over union of the detection bounding box and the ground truth annnotation) of at least 0.7.

In previous work the 3D annotation of object parts [7] or a set of 3D CAD models [12] of the class are needed. In [5] the pose of the object at test time as well as the camera intrinsic and extrinsic parameters are needed. In contrast,

in our method we only require a rough estimation of the cuboid containing the object (which can also be deduced from the bounding boxes aspect ratio, as explained in sect. 3) and the annotated pose of the object at training time. We can further reduce the amount of annotation to just the pose of one training sample. In this case we introduce in the training an additional iterative loop that, starting from a model trained only with the given sample for which we know the pose, incrementally selects a reduced set of high scoring samples and adds them to the training data. We repeat this until the entire training set is included. In this setting, as we do not know the viewpoint of the sample, we consider it as latent variable and let the model choose the pose that best explains the new sample. In the experiments we compare the results for the fully supervised setting with those of the weakly supervised setting here explained.

## 6. Experiments

We evaluate our model on two well known datasets for face and car detection and pose estimation. For cars we evaluate the fine-grained pose estimation on the EPFL car dataset [11]. As in [11], we use the first 10 videos for training and the other 10 for test. For faces as in [17] we train on 900 samples from MultiPIE [6] and we test on AFW [17]. We evaluate the viewpoint estimation performance in terms of pose estimation average precision (PEAP) initially introduced in [10], which represents the average precision of correct detections (with the VOC criterion) and correct pose estimation. To compare our method with other previous work we also evaluate the viewpoint estimation (VE) as the number of correctly estimated viewpoints normalized by the total number of objects in the dataset. We first compare our 3D HOG based model with similar approaches based on HOG templates on EPFL. Then we use AFW as test bed for several additional experiments and evaluations.

In table 1 we compare the performance of our 3D method with other HOG based methods in terms of detection and pose estimation on the EPFL car dataset. The performance of our 3D method is markedly better than [11], which is also a method based on rigid templates. Compared with deformable HOG templates [10] our method has lower performance. However, our current implementation of the 3D model does not consider deformable parts. We consider that introducing deformable parts in our 3D representation can further boost performance. We leave it as future work. In Fig. 3 we also show some examples of detection and viewpoint estimation on test images.

We compare our 3D method on AFW with our own implementation of a detector/view point estimation based on HOG templates, where each view is represented by a different rigid template. In this case the model based on HOG templates, as we use 13 views, is composed of 13 different templates, while our model has only the 3 faces of the cuboid representing the 1 frontal and 2 lateral views (the back of the head is not modeled). In terms of average precision in detection, HOG templates obtains a detection score of 79%, while our 3D model is very close with 78.8%. We also compare our methods with the implementation of HOG templates from [17] which obtains 75.5%. This indicates that our implementation is correct and slightly better tuned that the one in [17]. The state-of-the art on detection is 88.7% using a tree-structured model also from [17]. However that method is based on deformable models, while our approach uses a 3D representation, but still based on rigid HOG templates. For VE the implementation of HOG templates from [17] obtains 74.6% whereas our HOG template model obtain 78.4% and our 3D model 71.4%. Our 3D model is less accurate in viewpoint estimation than HOG templates. This is probably due to the fast approximation of the object appearance for non fronto-parallel views and to the reduced number of parameters to learn. Nevertheless, it is still quite impressive that with just 3 independent views we can obtain results comparable with a model with 13 different views. As we will see in the next subsection, using a much smaller model representation can also be an advantage in certain conditions.

| | Method | Det.(AP%) | PEAP(%) | VE(%) |
|---|---|---|---|---|
| 16 views | HOG templates [11] | 85 | - | 41.6 |
| | Def HOG templates [10] | 100 | 58 | - |
| | Def HOG templates Mod. [10] | 97 | 62 | 66.1 |
| | Our 3D model | 91.8 | 36.1 | 56.9 |
| 8 views | Def HOG templates [10] | 99 | 61 | - |
| | Def HOG templates Mod. [10] | 91 | 71 | 73.7 |
| | Our 3D model | 91.8 | 61 | 73.1 |

Table 1. Results on EPFL car dataset. Note that our 3D but rigid model obtains similar results to those obtained by models based on deformable parts.

### 6.1. Extrapolation of unseen views

An advantage of our method is the ability to detect objects from unseen viewpoints. To show that, we train our model with a limited number of viewpoints and compare its detection and viewpoint estimation performance with our own implementation of HOG templates, one for each view. Note that for this experiment the test set remains unaltered, i.e. including all viewpoints. As the number of training views is reduced, more and more of the test images will be observed from previously unseen viewpoints. We compare the performance of the two methods on 3 different configurations: 13 views, which is the standard multiPIE training composed of 900 samples, 7 views, where (-90,-60,-30,0,30,60,90) degrees angles are used and generate a training set of 600 samples and 4 views, where only (-90,-30,30,90) degrees views are used and compose a training set of 200 samples. In Fig. 4 (a) we show the average preci-

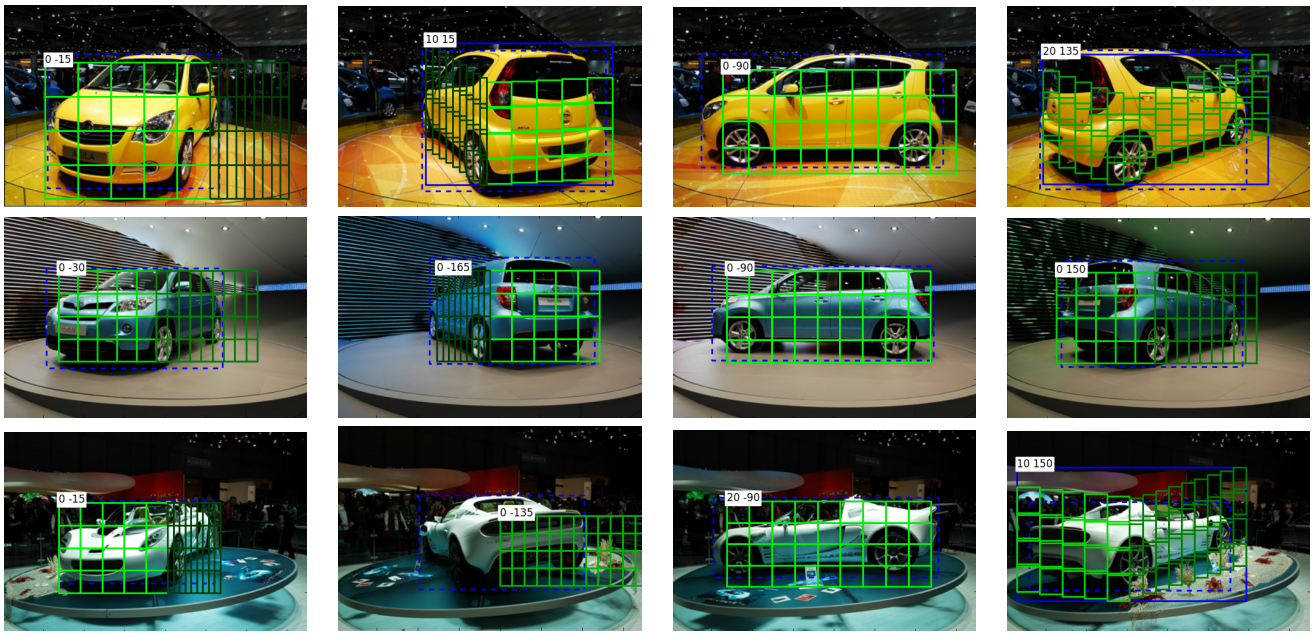



Figure 3. Examples of detection and viewpoint estimation on the EPLF car dataset. The green boxes represent the location of the HOG patches, the blue box is the final detection while the dashed blue box is the ground truth bounding box. The two numbers on top of the detection represent the detection viewpoint with the corresponding rotations on x and y axes.

sion and pose estimation of our 3D method and HOG templates while reducing the number of viewpoints. For the full training data, HOG templates are slower than our 3D approach but they obtain slightly better performance. Instead, when considering 4 views our method clearly outperforms the HOG templates.

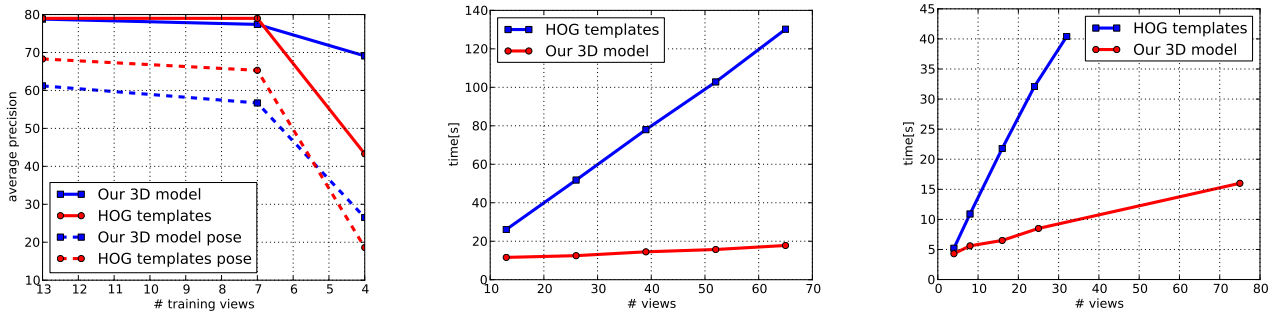
### 6.2. Computational cost

We compare the computational cost of our 3D representation and a detector based on a composition of HOG templates on AFW dataset and EPFL car dataset. On AFW we start with the standard configuration of 13 views which corresponds to a view each 15 degrees from -90 to 90 degrees on the $y$ axis. Now, if we want to model not only rotations on $y$, but also on $x$, the number of views is not separable in the two angles, therefore, the total number of views grows to 39 in case of modeling 3 views (-30,0,30 degrees on x) or to 65 if quantizing on 5 views. As shown in Fig.4 (b) the computational cost of the HOG template is proportional to the number of views. For our 3D model instead there is a constant cost due to the computation of the HOG frontoparallel patches and then the cost of increasing the number of views is very reduced. Notice that even when using a reduced set of views our method is still faster than the HOG templates because our basic model is based on 3 HOG templates whereas the basic model with HOG templates uses 13 HOG templates.

In Fig.4 (c) we show the time needed by HOG templates and our 3D model on EPFL cars. In this case, when using a model with 4 views, the two approaches have similar inference time. This is expected because on cars also our 3D model has to scan 4 views and therefore the computational time is similar. However when performing a much finer viewpoint estimation, the computational cost of the HOG templates becomes very high. For instance with 24 views (a view each 15 degrees spanning from 0 to 360 degrees on the $y$ axis) the inference time is more than 32s. In our model the inference time still grows linearly in the number of views, but much slower. For instance, when evaluating 24 views the inference time is less than 10 seconds. For cars it can be useful to estimate also the viewpoint on the $x$ axis. In this case, combining the 24 viewpoints on the $x$ axis with angles of $(0, 10, 20)$ degrees on the $y$ axis, produces 72 different views which can be computed in around 16 seconds with our model. This experiment shows how our 3D model is especially indicated for estimating fine viewpoints and multiple angles where the number of views can easily grow to more than 100 and using a set of HOG templates would be computationally unfeasible.

### 6.3. Weakly Supervised Learning

Often the viewpoint of the object is not available or just a rough guess of it is given at training time. In these situations we can still use our algorithm with the modifications explained in sect. 5. Here we test the weakly supervised learning for faces. As before we train on multiPIE and test on AFW. We initialize the model with a single frontal face. After that, at each iteration we select the 20 best scoring




(a) (b) (c)

Figure 4. (a) Average precision vs. number of viewpoints used for training. We evaluate detection (continuous line) and pose estimation (dashed line) average precision for HOG templates (red) and our 3D model (blue) on the AFW dataset. Our method can cope much better with unseen viewpoints at test time as becomes obvious when the number of training viewpoints is reduced. (b)-(c) Computational cost for HOG templates (blue) and our 3D model (red). A high number of views is important if we want to model more than one viewpoint angle on AFW (b) or fine angle variations on EPFL car (c).

samples (where the viewpoint is selected as latent variable to maximize the score) of the training data and add them to the training. We repeat the procedure until all the 900 samples of MultiPIE are used. If the procedure works properly we expect that the method places the new samples at the correct viewpoint and slowly fills in the entire 3D representation.

In Fig. 5 we show the evolution of the object model while adding more samples. At the beginning only a frontal face is known. Then, by slowly adding the "easiest" faces, the model is filled up. We also show the corresponding performance of the model for detection. The final average precision (71.5%) is not so far from the one obtained in the supervised case (78.8%). This means that the weakly supervised learning can effectively be used for learning 3D models with minimum amount of annotations. For pose estimation the behavior of the average precision curves is quite similar and leads to a maximum average precision of $50.8\%$ which is around 10 points below the average precision for the supervised case ($61.2\%$).

## 7. Conclusions

In this paper we have presented a new and computationally efficient model for 3D object class detection and viewpoint estimation. The model is based on a cuboid representation that allows the method to share information from different viewpoints. The evaluation of multiple viewpoints is efficient because after computing the score of frontal views, the other views are quickly approximated as linear combinations of parts of these. In this way we can scale up the estimation of more than 100 viewpoints in a reasonable time and with performance similar to HOG templates. This can be useful for a fine-grained viewpoint estimation, as well as when the estimation is computed over 2 angles, e.g. yaw and pitch. In future work we plan to allow the parts to deform as in deformable part models.

## Acknowledgments

This work was partially supported by the FP7 ERC Starting Grant 240530 COGNIMUND and the GOA KU Leuven project CAMETRON.

## References

[1] M. Arie-Nachimson and R. Basri. Constructing implicit 3d shape models for pose estimation. In *the Conference on Computer Vision and Pattern Recognition*, pages 1341–1348, 2009. 2

[2] C. Dubout and F. Fleuret. Deformable part models with individual part scaling. In *the British Machine Vision Conference*, pages 28.1–28.10, 2013. 3

[3] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 1

[4] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Transactions on Pattern Analysis and Machine Intelligence*, pages 1627–1645, 2010. 1, 2, 4

[5] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in Neural Information Processing Systems*, pages 611–619, 2012. 1, 2, 3, 4

[6] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 28(5), 2010. 5

[7] M. Hejrati and D. Ramanan. Analyzing 3d objects in cluttered images. In *Advances in Neural Information Processing Systems*, pages 602–610, 2012. 2, 4

[8] D. Hoiem, C. Rother, and J. M. Winn. 3d layoutcrf for multiview object class recognition and segmentation. In *the Conference on Computer Vision and Pattern Recognition*, 2007. 2

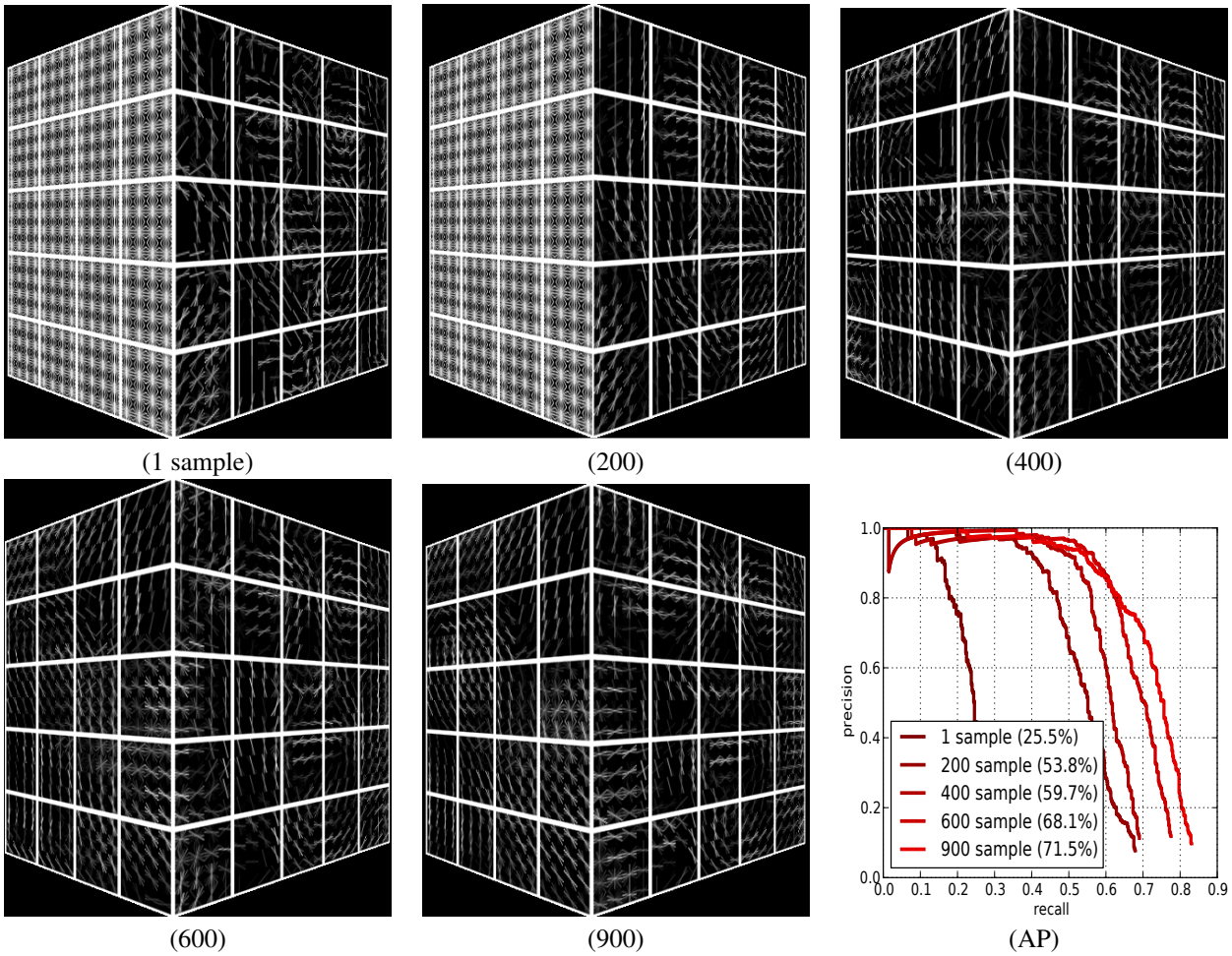



Figure 5. Evolution of the object model for weakly supervised training. In this settings we know the viewpoint of only one sample and then slowly all the other samples in the dataset are added (self-labelled) at the estimated viewpoint. Initially the lateral view of the model is empty. With 200 samples, the model still only represents frontal faces because in the dataset there are 300 frontal faces.


[9] A. M. Kushal and J. Ponce. A novel approach to modelling 3d objects from stereo views and recognizing them in photographs. In *Proceedings of the European Conference on Computer Vision*, 2006. 2

[10] R. Lopez-Sastre, T. Tuytelaars, and S. Savarese. Deformable part models revisited: A performance evaluation for object category pose estimation. In *the International Conference on Computer Vision, 1st Workshop on Challenges and Opportunities in Robot Perception*, 2011. 1, 2, 5

[11] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *the Conference on Computer Vision and Pattern Recognition*, 2009. 5

[12] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *the Conference on Computer Vision and Pattern Recognition*, 2012. 1, 2, 3, 4

[13] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. Learning a dense multi-vew representation for detection, viewpoint classification and synthesis of object categories. In *the International Conference on Computer Vision*, 2009. 2

[14] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. V. Gool. Towards multi-view object class detection. In *the Conference on Computer Vision and Pattern Recognition*, 2006. 2

[15] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial occulsion. In *Advances in Neural Information Processing Systems*, pages 1928–1936, 2009. 2

[16] Y. Xiang and S. Savarese. Estimating the aspect layout of object categories. In *the Conference on Computer Vision and Pattern Recognition*, 2012. 3

[17] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *the Conference on Computer Vision and Pattern Recognition*, pages 2879–2886, 2012. 5